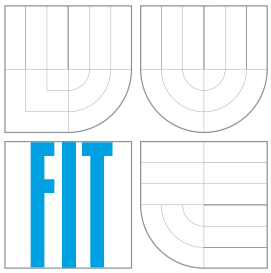
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# OVLÁDÁNÍ POČÍTAČE POMOCÍ GEST

CONTROLLING COMPUTER USING GESTURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER LACKO

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN JURÁNEK, Ph.D.

BRNO 2013

# Abstrakt

Tato práce se zabývá tvorbou systému pro ovládání počítače přes webkameru pomocí gest, přičemž gesto je zde chápáno jako pohyb ruky utvářející nějaký vzor. V úvodu jsou popsány metody pro detekci ruky v obrazu, sledování ruky a pro vyhodnocení pohybu použity v této práci. Následuje popis systému a jeho implementace, provádění a vyhodnocení testů. Výstupem práce je program umožňující jednoduché ovládání prohlížeče dokumentů a multimediálního přehrávače.



# Abstract

This work deals with creation of system for controlling computer through webcam with gestures. Gesture in this work can be viewed as hand motion forming some pattern. In the beginning are described methods for hand detection, hand tracking and pattern recognition. Afterwards comes description of system and it's implementation with tests evaluation. Outcome of this work is program for simple control of document viewer and multimedia player.



# Klíčová slova

ovládání počítače, gesta, Viola-Jones detektor, sledování, TLD, rozpoznávání vzorů, skryty markovovy modely, HMM, D-Bus

# Keywords

pc controlling, gestures, Viola-Jones detector, tracking, TLD, pattern recognition, HMM, D-Bus


# Citace

Peter Lacko: Ovládání počítače pomocí gest, bakalářská práce, Brno, FIT VUT v Brně, 2013

# Ovládání počítače pomocí gest

## Prohlášení

Prehlasujem že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Romana Juránka, Ph.D. a že som uviedol všetky literárne zdroje a publikácie z ktorých som čerpal.

.....
Peter Lacko
15. května 2013

## Poděkování

V úvode by som chcel poďakovať svojmu vedúcemu, pánovi Romanovi Juránkovi, za jeho odborné rady a pripomienky týkajúce sa tejto práce. Takisto ďakujem svojej rodine za ich podporu a všetkým tým, ktorí mi ochotne pomáhali so získavaním potrebných dát.

# Obsah


**1 Úvod** **2**

**2 Detekcia objektu v obraze detektorom Viola-Jones** **3**
2.1 Príznaky . . . . . 4
2.2 Algoritmus AdaBoost . . . . . 5
2.3 Trénovanie výsledného klasifikátoru . . . . . 5

**3 Sledovanie objektu metódou TLD** **9**
3.1 Sledovač (tracker) . . . . . 9
3.2 Detektor . . . . . 11
3.3 Integrátor . . . . . 13
3.4 Komponenta učenia . . . . . 14

**4 Rozpoznávanie gest pomocou Skrytých markovovych modelov** **15**
4.1 Left-Right modely . . . . . 16
4.2 Výpočet pravdepodobnosti sekvencie symbolov a trénovanie modelov . . . . 16
4.3 Extrakcia príznakov z gesta . . . . . 18
4.4 Generovanie sekvencií symbolov pre HMM . . . . . 18

**5 Návrh a implementácia systému** **20**
5.1 Implementácia a prepojenie komponent . . . . . 21

**6 Testovanie a výsledky** **24**
6.1 Detekcia ruky v obraze . . . . . 24
6.2 Rozpoznanie gest . . . . . 27

**7 Záver** **30**

**A Obsah CD** **33**

# Kapitola 1

# Úvod

Prístup k ovládaniu počítačových systémov zažíva v súčasnoti veľké zmeny a to najmä vďaka nástupu dotykových rozhraní prítomných v chytrých telefónoch, tabletoch, a čoraz častejšie i v notebookoch. Okrem dotykových rozhraní sa dostávajú do popredia i bezdotykové rozhrania, ktorých hlavnou výhodou je bezpochyby možnosť ovládania počítača z pohodlia kresla či venovanie sa iným činnostiam pri občasnej interakcii s počítačom – napríklad posun skladby v playliste. V súčasnosti najpoužívanejšie riešenia sú založené na použití špecializovaného hardwaru, akým je napríklad kinect[1]. Ich pomerne veľkou nevýhodou je ale nutnosť tento hardware vlastniť. Jediným podobným produktom pracujúcim výlučne s obrazom z webkamery je aplikácia Flutter[2], ktorá je ale dostupná len pre OS Windows a Macintosh.

Cieľom tejto práce je vytvoriť systém umožňujúci ovládanie vybraných programov cez webkameru a to pomocou jednoduchých pohybov ruky (gest). Systém má byť schopný pracovať len s použitím obyčajnej webkamery aké sú dostupné vo väčšine súčasných notebookov a má ho byť možné používať i pri umelom osvetlení.

Princíp činnosti je nasledujúci: Obraz z webkamery sa najprv testuje na prítomnosť zovretej ľudskej dlane. Po jej úspešnej detekcii je sledovaný jej pohyb a vyhodnotí sa trajektória tohoto pohybu. Ak bolo rozpoznané nejaké gesto, prevedie sa príslušná akcia. Príkladom vykonanej akcie je posun stránky v dokumente či spustenie/pozastavenie prehrávaného videa.

Druhá až štvrtá kapitola obsahujú popis metód použitých na detekciu objektu, sledovanie objektu a rozpoznanie gest. V kapitolách päť a šesť je popísaný návrh systému a jeho implementácia. Náplňou siedmej kapitoly je popis dátových sád a metód použitých pre testovanie detektoru a rozpoznávača gest, pričom v závere každej sekcie sú zhrnuté výsledky týchto testov. Výstupom práce je systém umožňujúci základné ovládanie prehliadača dokumentov a multimediálneho prehrávača pomocou šiestich jednoduchých gest.

[1] http://www.microsoft.com/en-us/kinectforwindows/
[2] https://flutterapp.com/

# Kapitola 2

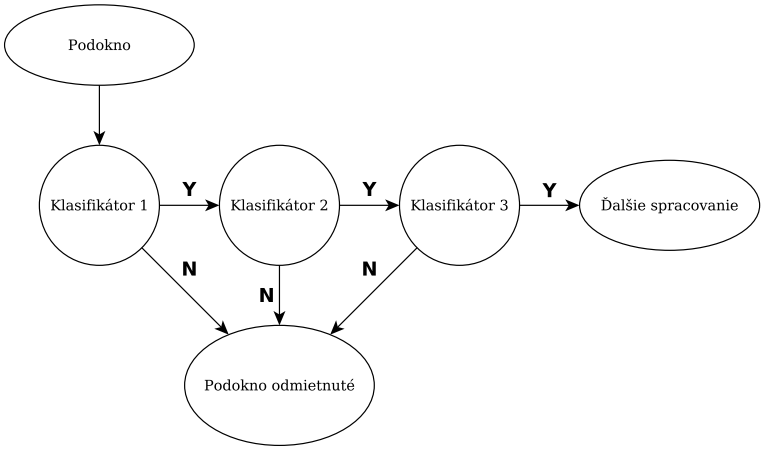
# Detekcia objektu v obraze detektorom Viola-Jones

Detekcia, alebo rozpoznanie objektu v obraze je jednou zo základných činností, ktorými sa odvetvie počítačového videnia zaoberá. Cieľom je zistiť či sa hľadaný objekt nachádza v obraze či sekvencii snímok, jeho polohu, veľkosť alebo iné vlastnosti.

Existuje mnoho metód detekcie objektov, pričom pre real-time detekciu je mimoriadne obľúbená kaskáda boostovaných klasifikátorov predstavená Violou a Jonesom [21] a to najmä vďaka svojej rýchlosti spracovania dát [19]. Metóda bola pôvodne určená k detekcii tvárí, ako ale uvádza Stojmenovic [20], rovnako dobre ju je možné využiť i na detekciu iných objektov, napr. áut alebo chodcov. V tejto práci je využitá na detekciu užívateľovej ruky v obraze z webkamery.

Vyhľadávanie objektu je realizované postupným posúvaním detekčného okna na všetky pozície v obraze, pričom veľkosť detekčného okna sa každým priechodom mení na základe hodnoty tzv. *scale faktoru*. Táto metóda detekcie sa nazýva aj sliding window (doslova kĺzajúce okno). Napríklad pri veľkosti detekčného okna $32 \times 32$ pixelov, veľkosti spracovávaného obrázku $320 \times 240$ a pri hodnote scale faktoru 1.1 je spracovávaných 802 856 podokien. Ak vezmeme do úvahy že prevažná väčšina skenovaných podokien hľadaný objekt neobsahuje, je žiadúce tieto podokná označiť za pozadie s čo najmenším vynaloženým úsilím — minimálnym možným počtom vykonaných inštrukcií. Toto je u Viola-Jones detektoru docielené zapojením niekoľkých jednoduchších klasifikátorov do série tak, aby prvý z týchto klasifikátorov zamietol čo najviac podokien s pozadím (a zároveň „prepustil“ všetky pozitívne). Takéto zapojenie je vlastne degenerovaný rezhodovací strom nazývaný kaskáda, schéma takejto kaskády je znázornená na obr. 2.1. Spracovávané podokno je označené za pozitívne len ak bolo označené za pozitívne všetkými klasifikátormi v kaskáde. Ak bolo podokno ktorýmkoľvek klasifikátorom označené ako pozadie, ďalej sa nespracováva.

Klasifikátory v jednotivých stupňoch kaskády sú natrénované pomocou algoritmu AdaBoost, pričom každý z týchto klasifikátorov využíva k svojej činnosti niekoľko slabých klasifikátorov. Tieto potom klasifikujú na základe hodnôt jedného alebo niekoľkých tzv. *príznakov*. Algoritmus AdaBoost, príznaky a algoritmus trénovania výsledného klasifikátoru sú popísané v nasledujúcich sekciách.




Obrázek 2.1: Kaskádové zapojenie klasifikátorov v detektore. Podokno je klasifikované ako pozitívne len vtedy, ake je všetkými klasifikátormi označené ako pozitívne.

## 2.1 Príznaky

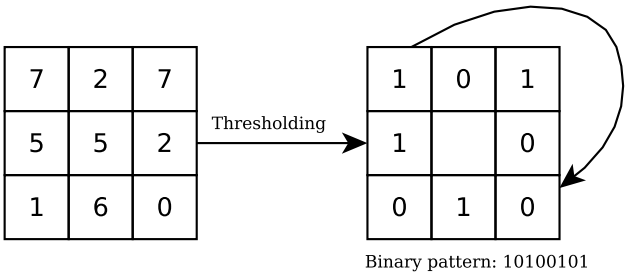
Príznaky môžeme chápať ako určité vizuálne vlastnosti objektu, ktoré sú preň charakteristické. Príznakom môže byť napr. hrana, tvar či farba pneumatiky na aute a pod. Vďaka príznakom sme schopní efektívne popísať objekt a následne ho spoľahlivo a rýchlo v obraze rozpoznať. Existuje niekoľko typov príznakov ktoré sme schopní z obrazu získať, pričom ich výber závisí predovšetkým na riešenom probléme – na rozponanie typu textúry využijeme iné príznaky ako na rozpoznanie chodca.

Príznaky môžeme rozdeliť do troch základných kategórií [10]: spektrálne, geometrické a príznaky textúry. Spektrálne príznaky využívajú hodnoty samostatných pixelov a ich distribúciu v obraze. Nedokážu síce zachytiť žiadne tvary ani spojenia medzi pixelmi, sú však nezávislé na veľkosti či deformáciách. Geometrické príznaky naopak zachycujú tvary objektov pričom ich výpočet je oproti príznakom prvého typu niekoľkonásobne rýchlejší. Ich príkladom sú napríklad gradientné metódy schopné popísať hrany a ich orientáciu, Haarove príznaky, alebo Local Binary Patterns, pričom práve posledné menovné sú využité v tejto práci. Príznaky tretieho typu slúžia na reprezentáciu homogénnych vzorov a ich vzťahu k okoliu.

### 2.1.1 Local Binary Patterns

Metóda LBP je založená na prahovaní okolia každého pixelu obrazu s hodnotou uprostred a následnom vytvorení binárneho reťazca (0 ak je hodnota pixelu menšia/rovná než hodnota uprostred, 1 ak je väčšia) [15], viď obr. 2.2. LBP ale slúži najmä na zachytenie mikroštruktúry vzorov v obraze a pri porovnávaní susedných pixelov je preto náchylná na lokálny šum a taktiež nedokáže popísať priestorovo väčšie oblasti obrazu [11].

Ich rozšírením vznikli Multi-Scale Block LBP ktoré vyššie spomínené nedostatky odstraňujú. MB-LBP používajú namiesto hodnôt jednotlivých pixelov priemernú hodnotu pixelov vo štvorcových oblastiach. Príznaky sa počítajú pre regióny rôznych veľkostí – napr. $3 \times 3$(„klasické“ LBP), $9 \times 9$, $15 \times 15$ atd. Vďaka tomu dokážu popísať mikro i makro štruktúry vzorov v obraze. MB-LBP sú teda oproti LBP robustnejšie, dokážu popísať i makroštruktúry vzorov v obraze a vďaka integrálnemu obrazu [4] je ich výpočet extrémne rýchly [11].




Obrázek 2.2: Prahovanie okolia $3 \times 3$ u LBP. Hodnoty okolitých pixelov sú prahované hodnotou pixelu v strede.

Príznaky najlepšie oddeľujúce pozitívne vzorky od negatívnych sa potom využijú v klasifikátore, pričom ich počet je závislý na riešenom probléme. V tomto prípade (detekcia ruky) využíva každý klasifikátor kaskády 3-10 príznakov, celkovo teda niekoľko desiatok MB-LBP príznakov.

## 2.2 Algoritmus AdaBoost

AdaBoost (Adaptive Boosting) je algoritmus strojového učenia predstavený Freundom a Schapirom [6] slúžiaci na skladanie klasifikátorov z jednoduchších klasifikátorov (najlepšie oddeľujúcich pozitívne data od negatívnych) a na natrénovanie silného klasifikátoru s cieľom minimalizácie chyby.

Jeho vstupom je ohodnotená množina obrázkov $(x_1, y_1) \ldots (x_n, y_n)$, pričom $y_i = 0$ pri negatívnych vzorkách resp. 1 u pozitívnych a jeho výstupom je silný klasifikátor určený váhovým súčtom slabých klasifikátorov. Každý vybraný príznak musí klasifikovať s pravdepodobnosťou $> 0.5$, tj lepšou než náhodnou. Nasledujúci vzťah popisuje závislosť slabého klasifikátoru $h$ na príznaku $f$.

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{pre } pf(x) < p\theta \\ 0 & \text{inak} \end{cases} \tag{2.1}$$

V (2.1) je $x$ spracovávané podokno súčasného snímku, $p$ parita udávajúca znamienko pre porovnanie a $\theta$ optimálne zvolený prah pre daný príznak. Veľkosť spracovávaného podokna je rovnaká ako aktuálna veľkosť detekčného okna.

Nech $T$ je počet hypotéz, každá vytvorená použitím jedného príznaku. Finálna hypotéza je tvorená váženou kombináciou $T$ hypotéz, pričom ich váhy sú nepriamo úmerné k trénovacím chybám. Algoritmus trénovania pomocou AdaBoost je potom uvedený v algoritme 1. Algoritmus bol prevzaný z [21]. Detailnejší popis metódy AdaBoost a jej variácií je možné nájsť v [6] a [5].

## 2.3 Trénovanie výsledného klasifikátoru

Výsledný klasifikátor pozostáva z niekoľkých klasifikátorov zapojených do kaskády. Pred samotným trénovaním sa zvolia požadované ciele: minimálna úspešnosť detekcie každého stupňa (pomer true positive a všetkých pozitívnych okien), false alarm rate(pomer false

**Algoritmus 1:** Výber príznakov a trénovanie klasifikátoru metódou AdaBoost.

- Vstupom je trénovacia množina obrázkov $(x_1, y_1), \ldots (x_n, y_n)$, kde $y_i$ je 0 alebo 1 pre pozitívne resp. negatívne vzorky.
- Inicializuj váhy $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ pre $y_i = 0, 1$, $m$ a $l$ je počet pozitívnych resp. negatívnych vzoriek.
- **for** $t = 1 \ldots T$ **do**

  1. Normalizuj váhy
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  2. Vyber najlepší slabý klasifikátor, tj. s najnižšou chybou.
  $\epsilon_t = \min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$.
  3. Definuj $h_t(x) = h(x, f_t, p_t, \theta_t)$ kde $f_t$, $p_t$ a $\theta_t$ minimalizujú $\epsilon_t$.
  4. Uprav váhy:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i},$$
  kde $e_i = 0$ ak je vzorka $x_i$ klasifikovaná správne, inak $e_i = 1$ a $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

  **end**

- Výsledný silný klasifikátor je daný vzťahom:
$$C(x) = \begin{cases} 1 & \text{pre } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0 & \text{inak} \end{cases}$$
kde $\alpha_t = \log \frac{1}{\beta_t}$.

positive a všetkým okien s pozadím) a počet klasifikátorov v kaskáde.
Celková úspešnosť detekcie $D$ je

$$D = \prod_{i=1}^{K} d_i, \tag{2.2}$$

kde $K$ je počet klasifikátorov a $d_i$ je úspešnosť detekcie každého z nich. Podobne celkový false alarm rate $F$ je

$$F = \prod_{i=1}^{K} f_i, \tag{2.3}$$

kde $f_i$ je false alarm rate každého klasifkátoru kaskády. Úspešnosť 0.9 môže byť dosiahnutá zapojením desiatich klasifikátorov, každý s úspešnosťou 0.99 ($0.99^{10} \approx 0.9$) a s false alarm rate na úrovni $30\%(0.3^{10} \approx 6 * 10^{-6})$ [21]. Očakávaný počet vyhodnocovaných príznakov na pozitívnych vzorkách je pravdepodobnostný proces závislý na úspešnosti detekcie jednotlivých klasifikátorov:

$$N = n_0 + \sum_{i=1}^{K} (n_i \prod_{j<i} p_j) \tag{2.4}$$

kde $N$ je očakávaný počet vyhodnotených príznakov, $K$ počet klasifikátorov, $p_i$ úspešnosť $i$-tého klasifikátoru a $n_i$ počet príznakov v $i$-tom klasifikátore.

Cieľom trénovania je minimalizovať očakávaný počet vyhodnocovaných príznakov $N$ pri dosiahnutí požadovanej úspešnosti detekcie $D$ a false alarm rate $F$. Každá vrstva kaskády je natrénovaná pomocou AdaBoostu, pričom sa postupne zvyšuje počet použitých príznakov, až kým nie je dosiahnutá požadovaná úspešnosť danej vrstvy. Trénovanie pokračuje kým nie je dosiahnutý daný počet stupňov kaskády. Sada negatívnych vzoriek je pre nasledujúce vrstvy vytváraná z falošných detekcií získaných testovaním detektoru na validačnej sade. Kompletný postup trénovania je uvedený v algoritme 2.

**Algoritmus 2:** Trénovanie kaskádového detektoru

- Užívateľ si najskôr zvolí hodnoty $f$ – maximálny akceptovateľný false alarm, $d$ – minimálnu úspešnosť detekcie jednej vrstvy kaskády a počet stupňov kaskády $C$.
- $P$ = množina pozitívnych vzoriek
- $N$ = množina negatívnych vzoriek
- $i = 0$
- **for** $i = 1 \ldots C$ **do**
  - $i \leftarrow i + 1$
  - $n_i = 0$, $F_i = 1.0$, $D_0 = 1.0$
  - **while** $F_i > f$ **do**
    - $n_i \leftarrow n_i + 1$
    - AdaBoostom pomocou $P$ a $N$ natrénuj klasifikátor s $n_i$ príznakmi
    - Zisti aktuálnu hodnotu $F_i$ a $D_i$ spustením detektoru nad validačnou množinou dát
    - Znižuj prah $i$tého klasifikátoru pokiaľ úspešnosť detekcie doposiaľ vybudovaného detektoru nie je aspoň $d$, čo samozrejme ovplyvní aj $F_i$

    **end**
  - $N \leftarrow \emptyset$
  - pridaj všetky nesprávne detekované vzorky do $N$

**end**

# Kapitola 3

# Sledovanie objektu metódou TLD

TLD (Tracking-Learning-Detection) je sledovací systém predstavený Kálalom [9], určený na dlhodobé sledovanie ľubovoľného objektu. Dokáže sa prispôsobovať zmenám veľkosti a natočenia objektu a taktiež ho reinicializovať po jeho zmiznutí a znovuobjavení sa. Toto je možné vďaka zapojeniu trackeru, detektoru a učiteľa do jedného celku tak, ako je znázornené na obr. 3.1. V tejto práci slúži na sledovanie ruky po jej detekcii detektorom, a bol využitý práve kôli schopnosti prispôsobovať sa zmenám objektu a rýchlemu spracovaniu dát.

Tracker odhaduje pohyb objektu v obraze medzi za sebou nasledujúcimi snímkami, pričom predpokladá jeho neustálu prítomnosť v obraze. Detektor považuje za sebou idúce snímky za nezávislé a prehľadáva celé okno v obraze s cieľom nájsť hľadaný objekt v podobe, v akej bol pozorovaný v minulosti. Poloha objektu je potom určená kombináciou výsledku trackeru a detektoru. Učiteľ vyhodnocuje beh trackeru i detektoru, odhaľuje chyby detektoru a generuje trénovacie vzorky pre detektor.

## 3.1 Sledovač (tracker)

Úlohou trackeru je zistiť polohu sledovaného objektu v obraze v čase $t + 1$ a to iba na základe znalosti jeho polohy v obraze v čase $t$. Poloha objektu je v obraze reprezentovaná *boundig boxom*, ktorý je v prvej snímke ručne inicializovaný. Po inicializácii je skonštruovaná množina bodov rovnomerne rozložených vnútri bounding boxu. Posun týchto bodov medzi predchádzajucou a súčasnou snímkou je odhadnutý pomocou metódy Lucas-Kanade [13, 2] slúžiacej na odhad optického toku v obraze. Nasleduje detekcia zlyhania metódou forward-backward [8] spoločne s výpočtom normalizovaného korelačného koeficientu (NCC).

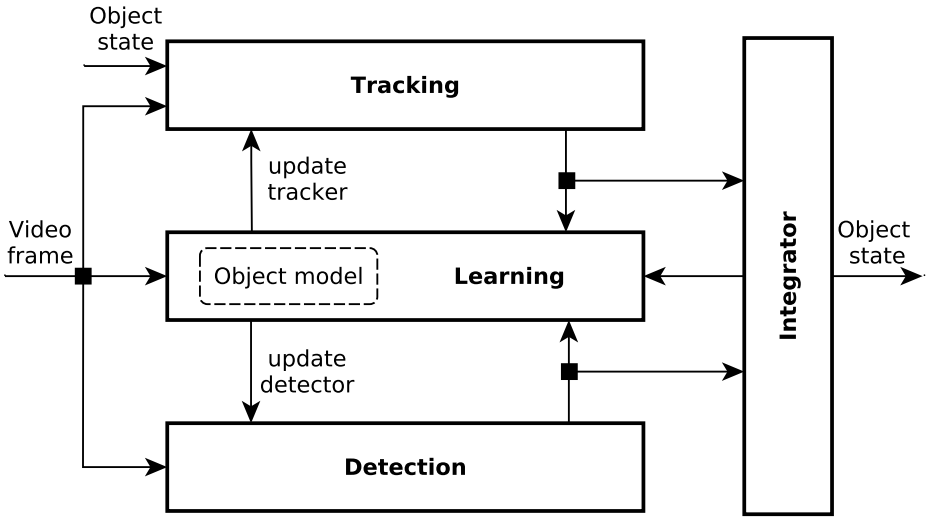
U detekcie zlyhania metódou forward-backward je každý bod $p$ v pôvodnom bounding boxe sledovaný v čase dopredu a späť metódou Lucas-Kanade, $p'' = LK(LK(p))$ a je vypočítaná chyba $\varepsilon = |p - p''|$. Principálne je metóda forward-backward znázornená na obr. 3.2.

Výpočet NCC je založený na podobnosti pôvodného *patchu* obsahujúceho body $p$ a patchu obsahujúceho výsledok sledovania, tj. body $p'$.

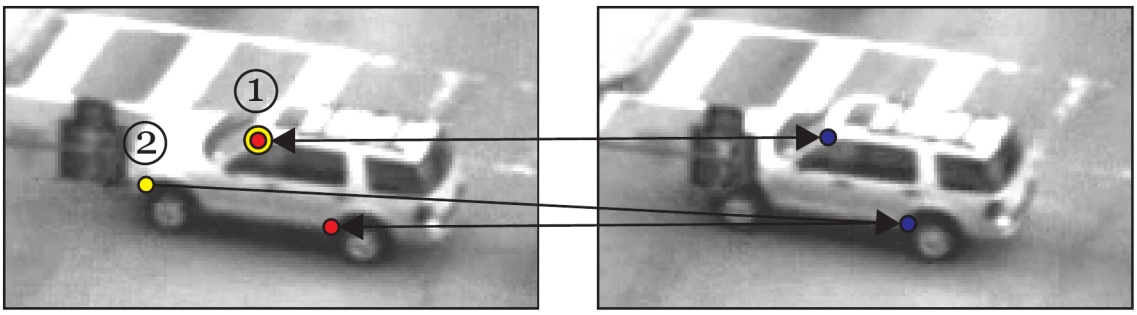
$$\mathrm{NCC}(P_1, P_2) = \frac{1}{n-1} \sum_{x=1}^{n} \frac{(P_1(x) - \mu_1)(P_2(x) - \mu_2)}{\sigma_1 \sigma_2} \tag{3.1}$$

$P_1$, $P_2$ sú porovnávané patche a $\mu_1$, $\mu_2$, $\sigma_1$ a $\sigma_2$ ich stredné hodnoty a rozptyly.

Za korektne vysledované body sa považujú iba body s chybou menšou ako medián chýb $med_{FB}$ a podobnosťou väčšou alebo rovnou $med_{NCC}$. Ak je navyše $med_{FB}$ väčší ako




Obrázek 3.1: Bloková schéma systému TLD. Prevzaté z [9].




Obrázek 3.2: Detekcia zlyhania metódou forward-backward. Trajektória bodu 1 je odhadnutá správne, avšak na druhej snímke je bod 2 zakrytý a sledovanie tohoto bodu preto zlyhá. Prevzaté z [8].

prah $\theta_{FB}$, sledovanie sa považuje za neúspešné. Zo zvyšných bodov sa získa nová poloha bounding boxu. Z pomerov relatívnych vzdialností medzi pôvodnými a novými bodmi sa určí zväčšenie/zmenšenie bounding boxu, posuny v osách x a y sa získajú rozdielom mediánov horizontálnych resp. vertikálnych súradníc bodov.

## 3.2 Detektor

Detektor umožňuje reinicializáciu trackeru, keďže na rozdiel od neho nie je závislý na polohe objektu v predchádzajúcej snímke. Detekcia je vykonávaná prístupom sliding window, kedy je detekčné okno posúvané obrazom a to v rôznych veľkostiach. Detektor samotný pozostáva zo štyroch jednoduchších detektorov, ktoré sú podobne ako v kapitole 2, zapojené do kaskády a testované podokno musí byť akceptované všetkými detektormi, aby bolo klasifikované ako pozitívne. Týmito detektormi sú *foreground detector* (detektor popredia), *variance filter* (variančný filter), *ensemble klasifikátor* a *nearest neighbour klasifikátor*.

### 3.2.1 Foreground detector

Táto metóda je založená na odčítaní pozadia, kde je každá snímka porovnávaná s modelom pozadia [14]. Proces vytvorenia modelu pozadia je mimo rozsah tejto práce, zaoberá sa ním napr Radke [18]. Po vytvorení pozadia sa od neho odčíta pôvodný obrázok, z ktorého následne prahovaním dostaneme binárny obrázok. Pomocou tzv. labeling algoritmu [14, 3] sú postupne označené všetky plochy potenciálne obsahujúce objekt. Následne sú zamietnuté všetky patche ktoré nie sú plne obsiahnuté v minimálnych boundig boxoch ohraničujúcich zvyšné komponenty (plochy). Ak nie je k dispozícii žiadne pozadie, sú týmto prvým detektorom akceptované všetky podokná.

### 3.2.2 Variance filter

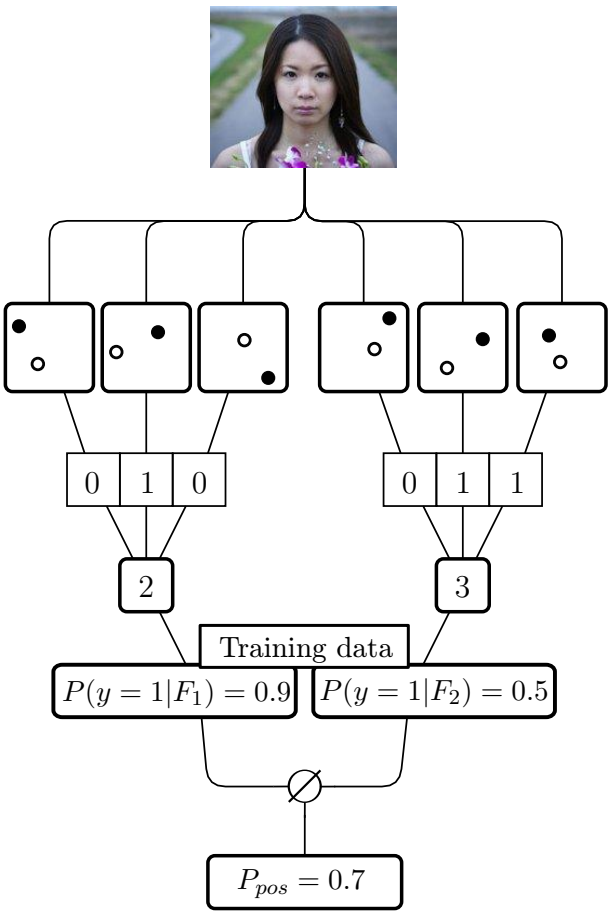
V tejto fáze je k odmietnutiu/akceptovaniu patchov využívaná podobnosť rozptylu hodnôt medzi skenovaným a vzorovým patchom, pričom sa odmietne približne polovica podokien [9]. Ak budeme považovať obrázok za jedno-dimenzionálny vektor hodnôt, jeho rozptyl $\sigma^2$ je definovaný [14] ako

$$\sigma^2 = \frac{1}{n} ii^2(B) - \left[\frac{1}{n} ii(B)\right]^2, \tag{3.2}$$

kde $ii(B)$ je súčet pixelov podokna $B$ a $ii^2(B)$ súčet ich druhých mocnín. Patche s hodnotou rozptylu nižšou ako je prah $\sigma^2_{min}$ sú v tejto fáze zamietnuté, kde $\sigma^2_{min}$ je rovný polovici rozptylu vzorového patchu.

### 3.2.3 Ensemble klasifikátor

Tento zložený klasifikátor obsahuje niekoľko *random fern* klasifikátorov [16, 14], klasifikujúcich na základe porovnania hodnôt niekoľkých náhodne vybraných pixelov v obraze. Princíp tejto metódy je zobrazený na obr. 3.3. Ak je hodnota pixelu v čiernom bode vačšia ako hodnota v bielom bode, hodnota príznaku $f_i = 1$, inak $f_i = 0$. Tieto príznaky sú zreťazené a ako skupina sa nazývajú fern. Fern môže nadobúdať maximálne $2^S - 1$ hodnôt, kde $S$ je počet príznakov (porovnávaných dvojíc) v jednom ferne.




Obrázek 3.3: Príklad klasifikácie pomocou dvoch trojpríznakových random ferns.

Hodnota fernu slúži ako index do polí obsahujúcich počet pozitívnych resp. negatívnych dát prislúchajúcich tejto hodnote. Pomocou hodnôt v poli sa potom výpočíta posteriórna pravdepodobnosť $P(y = 1|F)$, tj. pravdepodobnosť že patch je klasifikovaný ako pozitívny. Výpočet tejto pravdepodobnosti bude popísaný v sekcii 3.4. Výsledná pravdepodobnosť $P_{pos}$ je daná priemerom pravdepodobností ako

$$P_{pos} = \frac{1}{M} \sum_{k=1}^{M} P(y = 1|F_k) \tag{3.3}$$

kde $M$ je počet fernov použitých v klasifikátore. Hranica pre klasifikáciu je $P_{pos} = 0.5$.

### 3.2.4 Nearest neighbour klasifikátor

V poslednej fáze detekcie je podokno zmenšené na veľkosť 15x15 pixelov a porovnávané s každou vzorkou v modeli objektu a to pomocou normalizovaného korelačného koeficientu (NCC) uvedeného v sekcii 3.1, vzťah (3.1). Pomocou vzťahu (3.4) prevedieme výsledok z intervalu -1–1 do intervalu 0–1, vyjadrujúceho vzdialenosť dvoch patchov.

$$d(P_1, P_2) = 1 - \frac{1}{2}(\text{ncc}(P_1, P_2) + 1). \tag{3.4}$$

Ak si označíme množinu pozitívnych vzoriek ako $\mathcal{P}^+$ a množinu negatívnych vzoriek ako $\mathcal{P}^-$, minimálna vzdialenosť patchu $P$ k vzorke z pozitívnej triedy je

$$d^+ = \min_{P_i \in \mathcal{P}^+} \text{d}(P, P_i) \tag{3.5}$$

a k vzorke z negatívnej zasa

$$d^- = \min_{P_i \in \mathcal{P}^-} \text{d}(P, P_i). \tag{3.6}$$

Vzťah (3.7) vyjadruje dôveryhodnosť toho, že klasifikovaná vzorka patrí do množiny $\mathcal{P}^+$.

$$p^+ = \frac{d^-}{d^- + d^+} \tag{3.7}$$

Aby bolo podokno klasifikované ako pozitívne, musí $p^+$ byť väčšie ako prah $\theta^+ = 0.65$

Prah $\theta^+$ udáva hodnotu od ktorej sú podokná klasifikované ako pozitívne, podokná s dôveryhodnosťou nižšou ako $\theta^-$ sú klasifikované ako negatívne. Hodnota $\theta^-$ je stanovená na 0.5.

## 3.3 Integrátor

Úlohou integrátoru je zlúčenie výsledkov detektoru $D_t$ a sledovača $R_t$ do jedného bounding boxu $B_t$ a posúdenie dôveryhodnosti tohoto výsledku. Výstup integrátoru závisí na dôveryhodnosti výstupu detektoru $p^+_{D_t}$ a na dôveryhodnosti výstupu sledovača $p^+_{R_t}$, ktorý získame aplikovaním nearest neighbour klasifikátoru na jeho výstup.

Ak je výstupom detektoru práve jeden bounding box s dôveryhodnosťou vyššou ako je výstup sledovača, je sledovač reinicializovaný a do $B_t$ je uložený výstup detektoru. V prípade že je výstupom detektoru viac ako jeden pozitívny výsledok, alebo práve jeden avšak s dôveryhodnosťou nižšou než je výstup sledovača, je do $B_t$ uložený výstup sledovača. Vo všetkých ostatných prípadoch $B_t = \emptyset$.

Trénovanie tak ako je popísané v nasledujúcej sekcii sa prevedie len ak je výstup validný, tj. $p^+_{R_t} > \theta^+$ alebo ak bol predchádzajúci výstup validný a $p^+_{R_t} > \theta^-$. Inak sa výstup nepokladá za validný.

## 3.4 Komponenta učenia

### 3.4.1 Model objektu

Model objektu [9] je dátová štruktúra obsahujúca pozitívne a negatívne vzorky dát, tj. tie obsahujúce objekt resp. pozadie. Na základe podobnosti s týmito vzorkami sú potom ohodnocované nové patche.

Po inicializácii sú pozitívne patche získané transformáciami a posunmi bounding boxu, vďaka čomu je objekt možné detekovať pod rôznymi uhlami. Negatívne patche sú potom získané z pozadia. Podmienky pre pridanie nových vzoriek do modelu objektu sú popísané v nasledujúcej sekcii.

### 3.4.2 P-N learning

Učiteľ má za úlohu inicializovať detektor a v čase zlepšovať jeho schopnosti, k čomu sa využívajú dva typy *expertov* – P-expert a N-expert. Patrí medzi metódy semi-supervised learning, tj. k učeniu sú využívané ako ohodnotená tak neohodnotená množina dát. Úlohou expertov je odhadnúť detektorom chybne klasifikované patche a zmeniť ich označenie.

**P-expert** analyzuje patche klasifikované ako negatívne, odhadne tie nesprávne klasifikované a pridá ich do trénovacej množiny s kladným ohodnotením. Jeho odhad je založený na predpoklade, že všetky boundig boxy ktoré sa vysoko prekrývajú s $B_t$ ($> 60\%$) musia byť klasifikované ako pozitívne.

**N-expert** analyzuje vzorky klasifikované ako pozitívne, odhadne tie nesprávne klasifikované a pridá ich do trénovacej množiny so záporným ohodnotením. Odhad N-experta sa zakladá na tom, že objekt sa nachádza len na jednom mieste v obraze. Ak je teda prekrytie boundig boxu $B$ s $B_t$ minimálne alebo žiadne ($< 20\%$), bounding box musí byť klasifikovaný ako pozadie.
Obaja experti sa dopúšťajú chýb, ale vďaka ich vzájomnej nezávislosti sa tieto chyby kompenzujú [9].

Pravdepodobnosť, že patch je klasifikovaný ako pozitívny pri danej hodnote fernu $F_k$ je daná nasledovne

$$P(y = 1|F_k) = \begin{cases} \frac{p_{F_k}}{p_{F_k}+n_{F_k}}, & \text{ak } p_{F_k} + n_{F_k} > 0, \\ 0, & \text{ak } p_{F_k} + n_{F_k} = 0. \end{cases} \tag{3.8}$$

Vo vzťahu (3.8) je $p_{F_k}$ počet aktualizácií vykonaných P-expertom pre túto hodnotu fernu, a $n_{F_k}$ počet aktualizácií vykonaných N-expertom.

Proces učenia — výpočtu a aktualizácie fernov a pridávania nových patchov do modelu objektu — je nasledujúci. Vstupom algoritmu učenia je aktuálne spracovávané okno a bounding box $B_t$, udávajúci polohu objektu v ňom. Pre každý patch v obraze sa vypočíta jeho prekryv s bounding boxom a jeho dôveryhodnosť. Ak je prekryv $> 60\%$ a dôveryhodnosť $< 0.5$ (false negative), vypočítajú sa hodnoty fernov pre tento patch a aktualizuje sa $p_{F_k}$. Ak je naopak jeho prekryv $< 20\%$ a zároveň dôveryhodnosť $> 0.5$ (false positive), aktualizuje sa $n_{F_k}$ a ak je navyše i $p^+_{B_t} > \theta^-$, patch sa pridá do množiny negatívnych vzoriek $\mathcal{P}^-$.

Do $\mathcal{P}^+$ sa nakoniec pridá patch ohraničený bounding boxom $B_t$, avšak len v prípade, že jeho dôveryhodnosť je nižšia ako $\theta^+$, tj. $p^+_{B_t} < \theta^+$. Stojí za pripomenutie, že uvedený proces učenia prebieha len v prípade že výsledný bounding box je považovaný za validný.

# Kapitola 4

# Rozpoznávanie gest pomocou Skrytých markovovych modelov

Skrytý markovov model (HMM) je dvojitý stochastický proces ktorý vznikol rozšírením markovoveho procesu o prípady, kedy je pozorovaná veličina pravdepodobnostnou funkciou stavu. U HMM proces nie je priamo pozorovateľný (= je skrytý) a môže byť pozorovateľný iba pomocou iného stochastického procesu emitujúceho sekvenciu symbolov. HMM teda pozostáva z množiny stavov a množiny symbolov, ktoré tieto stavy generujú.

V tejto práci sú skryté markovove modely použité na priradenie užívateľom vykonaného gesta k jednému zo šiestich vzorov (viď použité gestá v kapitole kapitole 5), inými slovami na rozpoznanie gesta. Modely sú v tejto kapitole opísané tak, ako boli predstavené Rabinerom [17].

Formálne je HMM definovaný päticou $N, M, A, B, \pi$ kde:

- $N$ je počet skrytých stavov v modeli. Množinu stavov označíme ako $S = \{S_1, S_2, \ldots, S_N\}$ a $q_t$ je stav v čase $t$.
- $M$ je počet pozorovateľných symbolov korešpondujúcich s fyzickým výstupom modelovaného systému. $V = \{V_1, V_2, \ldots, V_M\}$ je potom množina týchto symbolov.
- $A = \{a_{ij}\}$ je pravdepodobnostné rozdelenie prechodov medzi stavmi, alebo prechodová matica, kde

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad i, j = 1, \ldots, N, \tag{4.1}$$

$$\sum_{j=1}^{N} a_{ij} = 1, \quad i = 1, \ldots, N. \tag{4.2}$$

  Ak je možné z každého stavu dosiahnuť jediným prechodom ľubovoľný stav, potom $a_{ij} > 0$ pre všetky $i, j$, v opačnom prípade $a_{ij} = 0$ pre jednu alebo viac kombinácií $i, j$.

- $B = \{b_j(k)\}$ je rozdelenie pravdepodobnosti pozorovaných symbolov v stave $j$, kde

$$b_j(k) = P[V_k \text{ v čase } t | q_t = S_j], \quad j = 1, \ldots, N, k = 1, \ldots, M, \tag{4.3}$$

$$\sum_{k=1}^{M} b_j(k) = 1, \quad j = 1, \ldots, N. \tag{4.4}$$

- $\pi = \{\pi_i\}$ je rozdelenie pravdepodobnosti počiatočných stavov, kde

$$\pi_i = P[q_1 = S_i], \quad i = 1, \ldots, N, \tag{4.5}$$

$$\sum_{i=1}^{N} \pi_i = 1, \quad i = 1, \ldots N. \tag{4.6}$$

Takto definovaný model môžeme skrátene zapisovať ako $\lambda = (A, B, \pi)$.

Typickým príkladom HMM je hádzanie dvoma neférovými mincami, kedy máme k dizpozícii iba výsledky hodov a ostáva nám skryté ktorou mincou bolo hádzané. V takomto modeli sú dve mince reprezentované dvoma skrytými stavmi. Každý stav je charakterizovaný rozdelením pravdepodobností pre dve strany mince — hlavu a znak, tj. pozorované symboly — a prechody medzi stavmi sú charakterizované prechodovou maticou.

## 4.1 Left-Right modely

Left-right (LR HMM), alebo ľavo-pravý model je špeciálnym typom HMM, ktorého základnou vlastnosťou je

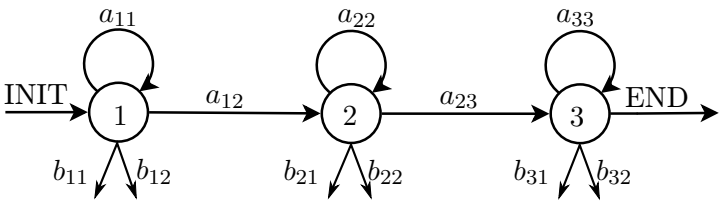
$$a_{ij} = 0, \quad j < i, \tag{4.7}$$

tzn. že sú povolené len prechody do stavov, ktorých index je rovnaký alebo väčší ako index súčasného stavu. Sekvencia stavov musí naviac začínať v stave 1. Okrem týchto podmienok sa často využíva i obmedzenie počtu stavov, do ktorých je možné prejsť:

$$a_{ij} = 0, \quad j > i + \Delta. \tag{4.8}$$

Pre $\Delta = 2$ môžeme zo stavu 1 prejsť len do stavov $1, 2$ a $3$.

Práve LR HMM sa najčastejšie využíva na rozpoznávanie rôznych vzorov – či už v obraze alebo zvuku. Napríklad Yoon [22] alebo Hu [7] využili LR modely pre rozpoznávanie ručne písaných znakov. V tejto práci je využitý taktiež tento typ modelu, pričom hodnota $\Delta$ je vždy rovná jednej, viď obr. 4.1.




Obrázek 4.1: Príklad LR modelu s troma stavmi, dvoma emitovanými symbolmi a $\Delta = 1$.

## 4.2 Výpočet pravdepodobnosti sekvencie symbolov a trénovanie modelov

Pre použitie skrytých markovovych modelov bolo potrebné:

1. Pri danej sekvencii symbolov $O = O_1 O_2 \ldots O_T$ a modele $\lambda = (A, B, \pi)$, efektívne zistiť $P(O|\lambda)$, tj. pravdepodobnosť že sekvencia $O$ bola vygenerovaná modelom $\lambda$, resp. nakoľko mu táto sekvencia odpovedá.

2. Pri danej sekvencii pozorovaní $O = O_1 O_2 \dots O_T$, zistiť hodnoty parametrov $A, B, \pi$ tak, aby sme maximalizovali $P(O|\lambda)$. Na tento problém sa dá pozerať aj ako na trénovanie HMM.

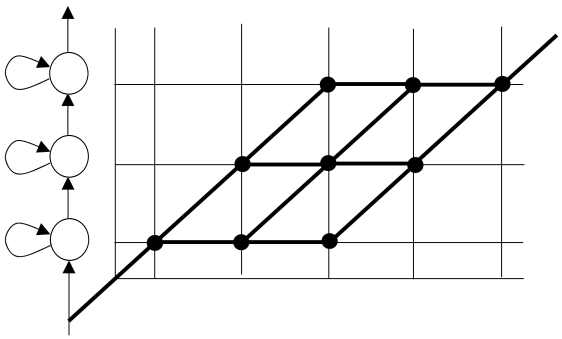
### 4.2.1 Výpočet pravdepodobnosti sekvencie symbolov

Sekvencia symbolov $O$ v našom prípade reprezentuje jednu realizáciu gesta získaného od užívateľa. Za predpokladu že máme natrénovaných niekoľko modelov schopných rozpoznať jednotlivé gestá, chceme pre každý takýto model $\lambda$ zistiť pravdepodobnosť $P(O|\lambda)$ („podobnosť“ gesta s gestom ktoré daný model rozpoznáva).

Na obr. 4.2 sú znázornené všetky cesty, ktorými môže model pri vyhodnocovaní sekvencie prejsť. Požadovanú pravdepodobnosť sekvencie symbolov pre daný model potom získame ako súčet spoločných pravdepodobností sekvencie a každej tejto cesty ako

$$P(O|\lambda) = \sum_{Q} P(O, Q|\lambda), \tag{4.9}$$

kde $Q$ označuje sekvenciu skrytých stavov (cestu). Procerúra pre efektívny výpočet tejto pravdepodobnosti sa nazýva *forward-backward* algoritmus [17].



Obrázek 4.2: Všetky cesty ktorými môže takýto LR model prejsť.

### 4.2.2 Trénovanie modelov

Trénovaním modelu sa snažíme nájsť také parametre $(A, B, \pi)$, ktorými maximalizujeme $P(O|\lambda)$ pre všetky trénovacie sekvencie. Týmito trénovacími sekvenciami sú vopred získané gestá prevedené na sekvenciu symbolov (viď sekcia 4.4). Počet sekvencií potrebných pre natrénovanie každého modelu je niekoľko desiatok až stoviek. Výstupom trénovania je napokon model, ktorý bude (v ideálnom prípade) všetky neznáme gestá *podobné* tým z trénovacej množiny, klasifikovať s vyššou pravdepodobnosťou ako všetky ostatné gestá.

Samotné trénovanie modelu je realizované algoritmom *Baum-Welch*, ktorý v každej iterácii vypočíta na základe hodnôt starých parametrov ich nové hodnoty. Bolo dokázané, že každým ďalším výpočtom sa pravdepodobnosť $P(O|\lambda)$ zvyšuje alebo nemení [17]. Trénovanie tak pokračuje kým rozdiel pravdepodobností medzi iteráciami neklesne pod určitú hodnotu, alebo nebol dosiahnutý maximálny daný počet iterácií. Keďže neexistuje exaktná metóda pre nastavenie počiatočných parametrov modelu, počiatočné hodnoty parametrov sú dané uniformným rozložením rešpektujúc (4.6),(4.2) a (4.4)

## 4.3 Extrakcia príznakov z gesta

Gesto je v počítači reprezentované vektorom $\mathcal{V}$ dvojíc $(x, y)$ obsahujúcim trajektóriu objektu v čase $t = 1, \ldots, T$. Z tohto vektoru sú následne pre každú dvojicu extrahované príznaky, vhodné pre efektívnu reprezentáciu vzoru za účelom jeho rozpoznania. Týmito príznakmi sú x-y príznaky a príznak rýchlosti (velocity), teda rovnaké ako využil Yoon [22].

Pomocou x-y príznakov je možné zakódovať informáciu o polohe objektu v obraze vrámci jedného gesta a to nezávisle na jeho veľkosti a posunutí. Jediným kritériom pre jeho rozpoznanie je teda jeho tvar. x-y príznaky sa pre každú dvojicu z $\mathcal{V}$ získajú nasledujúcim spôsobom. Nech $x_{min}, x_{max}, y_{min}$ a $y_{max}$ sú minimálne, resp. maximálne hodnoty $x$ a $y$, a $d_{max}$ nadobúda väčší z rozdielov korešpondujúcich si hodnôt. Potom príznak $f_t^x$ môže byť výrataný ako (4.11).Obdobne postupujeme i pre príznaky $f_t^y$, viď (4.12). Výsledná hodnota príznakov $f_t^x$, $f_t^y$ je potom v rozsahu 0-1 pre všetky $t = 1 \ldots T$.

$$d_{max} = \begin{cases} x_{max} - x_{min} & \text{ak} \quad x_{max} - x_{min} > y_{max} - y_{min}, \\ y_{max} - y_{min} & \text{inak} \end{cases} \tag{4.10}$$

$$f_t^x = \frac{x_t - x_{min}}{d_{max}}, \qquad t = 1, \ldots, T, \tag{4.11}$$

$$f_t^y = \frac{y_t - y_{min}}{d_{max}}, \qquad t = 1, \ldots, T, \tag{4.12}$$

$$f_t^v = \frac{v_t}{v_{max}}, \qquad t = 1, \ldots, T. \tag{4.13}$$

U príznaku rýchlosti, $f_t^v$, vychádzame z predpokladu, že rôzne gestá sa môžu kreliť rôznou rýchlosťou. Príznak rýchlosti teda vyjadruje vzdialenosť medzi dvoma bezprostredne po sebe nasledujúcimi bodmi. Vzťah pre jeho výpočet je uvedený v (4.13), kde $v_t$ je vzdialenosť medzi súčasným a nasledujúcim bodom a $v_{max}$ je maximálna vzdialenosť medzi dvoma bodmi. Hodnota príznaku rýchlosti je taktiež v rozmedzí 0-1.

## 4.4 Generovanie sekvencií symbolov pre HMM

Aby sme získali diskrétne symboly použiteľné v HMM, sú vektory príznakov extrahované z trénovacích sekvencií kvantizované. Na kvantizáciu vektorov príznakov je použitý algoritmus *k-means clustering* [1], ktorým vytvoríme $L$ clustrov v priestore. Clustre sú vytvorené tak, aby bola minimalizovaná vzdialenosť medzi vektormi v clustri a strednou hodnotou clustru (*mean*). Následne je strednej hodnote každého clustru priradené unikátne číslo (pozorovateľný symbol pre HMM) a spolu sú uložené do *codebooku*. Úplná procedúra vytvorenia codebooku je nasledovná:

1. Nech $L$ je počet symbolov v codebooku. Vyber náhodne $L$ vektorov z trénovacej množiny a prehlás ich za stredy clustrov.
2. Opakuj pre všetky trénovacie sekvencie:
   - Priraď každý element (vektor príznakov) trénovacej sekvencie do clustru s najnižšou vzdialenosťou od jeho stredu.
   - Aktualizuj codebook vyrátaním nových stredov clustrov.
3. Ukonči procedúru, ak sa obsah clustrov v dvoch po sebe idúcich iteráciách nemení, alebo bol dosiahnutý maximálny počet iterácií. Inak pokračuj bodom 2.

Každá sekvencia pre HMM — trénovacia, alebo neznáma — je potom vytvorená pomocou codebooku, pričom z codebooku sú vyberané symboly ktorým prislúchajú vektory príznakov s najmenšou vzdialenosťou od klasifikovaného vektoru.

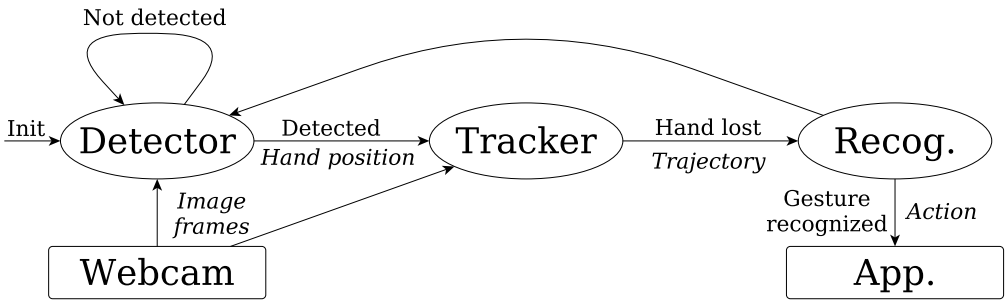
# Kapitola 5

# Návrh a implementácia systému

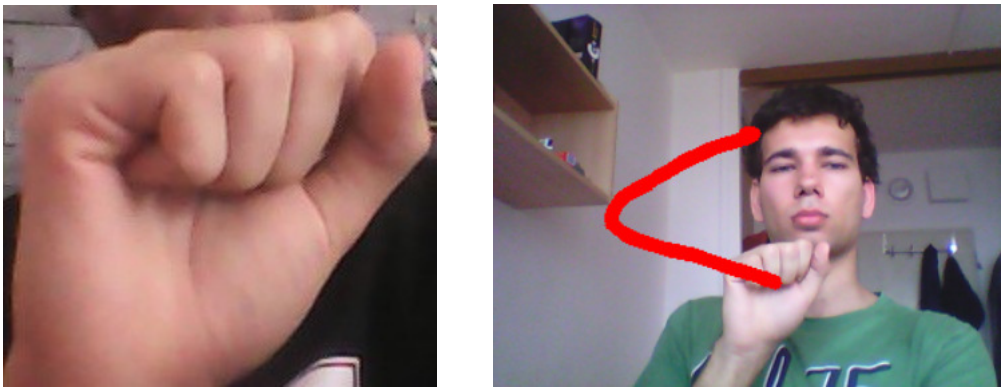
Systém ako celok sa skladá z troch základných komponent – detektoru, trackeru a komponenty pre rozpoznanie gest. Vstupom systému sú snímky získané z webkamery. V týchto snímkach detektor vyhľadáva ruku a v prípade úspešnej detekcie predá jej pozíciu v obraze a veľkosť trackeru. Ten následne zaháji sledovanie ruky a ukladá si jej polohu v každej spracovanej snímke, čím získa trajektóriu objektu v čase. Po zmiznutí ruky z obrazu odovzdá trajektóriu rozpoznávaču gest, ktorého úlohou je rozhodnúť ktoré z dopredu natrénovaných gest najlepšie odpovedá tejto trajektórii. V prípade zhody s niektorým gestom program prevedie akciu priradenú tomuto gestu, v opačnom prípade sa nedeje nič. Program potom prejde do počiatočného stavu, teda fáze detekcie ruky. Prepojenie týchto troch komponent je znázornené na obr. 5.1.

Aby bola ruka detektorom detekovaná, musí byť použitá pravá ruka zovretá do päste a otočená prstami ku kamere, viď obr. 5.2a. V pôvodnom návrhu som počítal s otvorenou dlaňou, ukázalo sa však že tracker dokáže presnejšie sledovať päsť. Príčinou je najmä prílišná jednotvárnosť dlane ale taktiež i jej väčšia plocha, čoho následkom je väčšia pravdepodobnosť že pri vykonávaní gesta opustí časť ruky zorné pole webkamery čím sledovanie objektu zlyhá.

Gestá ktorými sa systém ovláda sú v tejto práci chápané ako pohyby ruky utvárajúce nejaký vzor. Ukážka takéhoto gesta je na obr. 5.2b. V systéme využívam šesť typov gest – šípky vpravo, vľavo, nahoru, dolu, kruh a písmeno „S“. Ich tvary a povinné smery prevedenia sú znázornené na obr. 5.3. Tieto gestá boli vyberané tak, aby boli ľahko a rýchlo vykonateľné, poľahky zapamätateľné a aby ich bolo od seba možné jednoznačne odlíšiť. Zároveň ale gestá nesmú byť príliš jednoduché, aby sa po prípadnej falošnej detekcii detektoru a následnom prevední náhodného pohybu nevykonala nežiadúca akcia.

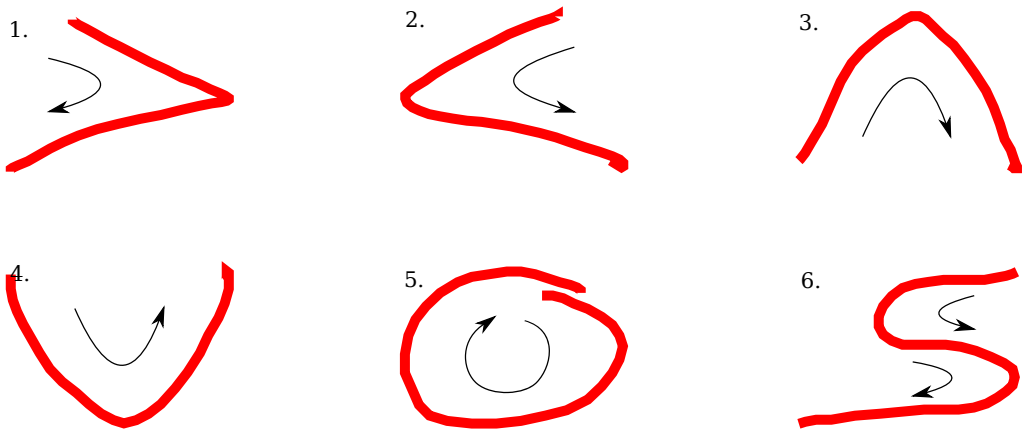



Obrázek 5.1: Bloková schéma systému pre ovládanie počítača pomocou gest.



(a) (b)

Obrázek 5.2: (a) Ukážka ruky v tvare, v akom je detekovaná, (b) Príklad gesta.




Obrázek 5.3: Šesť typov použitých gest: šípka vpravo, vľavo, nahoru, dolu, kruh a písmeno „S“.

## 5.1 Implementácia a prepojenie komponent

Aby bolo možné program používať i pri iných bežných činnostiach na pc, musí byť jeho využitie systémovych zdrojov minimálne. Z tohoto dôvodu bol ako implementačný jazyk celého systému zvolený jazyk C/C++. Z rovnakého dôvodu sú veľkosti všetkých spracovávaných obrázkov zmenšené na $320 \times 240$ pixelov, pričom táto veľkosť je zvolená ako vhodný kompromis medzi rýchlosťou spracovávania a maximálnou vzdialenosťou ruky od kamery a na správnu činnosť systému nemá výraznejší vplyv.

**Detektor.** Na natrénovanie a implementáciu detektoru som využil knižnicu OpenCV. OpenCV (Open Source Computer Vision Library) je voľne dostupná knižnica poskytujúca pohodlné rozhranie aplikáciám z oblasti počítačového videnia a strojového učenia[1].

Kaskáda klasifikátorov pozostáva z 18 stupňov, originálna veľkosť detekčného okna je $32 \times 32$ px. a hodnota scale faktoru $s = 1.1$. Detekcia je vykonávaná u každej dvanástej snímky, pri úspešnej detekcii sú spracované i štyri nasledujúce snímky. Pozitívna odozva detektoru nastane len v prípade že bola ruka detekovaná vo všetkých piatich snímkach

[1] viď http://opencv.org

Obrázek 5.4: Zmena veľkosti bounding boxu pri inicializácii trackeru. Červeným obdĺžnikom je označený výstup detektoru, modrým bounding box, ktorým je inicializovaný tracker.

a to v okolí max. 30 pixelov od prvej detekcie. Týmito obmedzeniami sa ešte viac zníži záťaž systému a zároveň sa predíde nežiadúcej detekcii, ktorá by mohla vzniknúť napríklad pri natáčaní webkamery. Následne sa inicializuje tracker s rozmermi a pozíciou detekčného okna získaného z detektoru, avšak zmenšeným o 20%, tak ako je znázornené na obr. 5.4. Dôvodom tohoto kroku je časté zlyhávanie trackeru pri prítomnosti veľkej časti pozadia v bounding boxe.

**Tracker.** Ako tracker som využil voľne dostupnú implementáciu systému TLD od Nebehaya – OpenTLD [14]. Tracker ukladá informáciu o pozícii ruky v každej snímke (ak je pozícia známa) do vektoru, pozícia je reprezentovaná dvojicou hodnôt $x, y$ označujúcich stred bounding boxu. Vektor s trajektóriou je predaný rozpoznávaču gest po 120 spracovaných snímkach alebo ak tracker stratí informáciu o polohe ruky vo ôsmich za sebou idúcich snímkach. Tieto hodnoty nie sú pre beh systému kritické. Prvá určuje maximálnu dĺžku trvania gesta, druhá potom odozvu systému na prevedené gesto.

**Rozpoznávanie gest.** Rozpoznávanie gest je realizované pomocou HMM, kde každé gesto je reprezentované jedným modelom (celkovo teda 6 modelov). Pre neznáme gesto je každým modelom vypočítaná pravdepodobnosť toho, že gesto bolo „generované“ práve týmto modelom. Neznáme gesto je potom klasifikované ako gesto s indexom modelu s najvyššou pravdepodobnosťou, alebo je nerozpoznané. Pre trénovanie všetkých modelov som stanovil maximálny počet iterácií algoritmu k-means pri tvorbe codebooku na 300 a maximálny počet iterácií Baum-Welch algoritmu na 20.

Na výpočet pravdepodobností a trénovanie modelov som využil voľne dostupnú implementáciu HMM od Dekang Lina[2]. Na zapuzdrenie práce s modelmi bola vytvorená trieda `Xhmm`, poskytujúca metódy na vytvorenie sekvencií symbolov z trénovacích dát a následné trénovanie modelov, vytváranie rozpoznávaných sekvencií symbolov z gest získaných z webkamery a taktiež na výber najlepšieho modelu.

**Prepojenie s ovládanou aplikáciou.** Aby bolo možné ovládať rozličné programy, bolo nutné použiť rozhranie, ktoré umožňuje jednoduchú komunikáciu medzi užívateľskými procesmi. Takýmto rozhraním je v Linuxe D-Bus. D-Bus je systém umožňujúci jednoduchú komunikáciu medzi procesmi na systémovej i užívateľskej úrovni. Funguje na princípe zasielania správ medzi procesmi: poskytuje procesom kanál, ktorým si môžu navzájom posielať správy, pričom každý proces má definovaný zoznam ponúkaných služieb (takýto pohľad je

[2] `http://webdocs.cs.ualberta.ca/~lindek/hmm.htm`, Copyright © 2003 Dekang Lin, lindek@cs.ualberta.ca

síce zjednodušený, pre pochopenie princípu však postačujúci). Detailný popis systému D-Bus je možné nájsť na domovskej stránke projektu[3]. Samotné napojenie na D-Bus bolo realizované využitím frameworku Qt4[4].

Pomocou tejto aplikácie je tak možné ovládať ľubovoľný program komunikujúci prostredníctvom D-Busu. Pokusne boli pripojené dva programy: prehliadač dokumentov okular a multimediálny prehrávač vlc. Popis ovládania týchto programov gestami je uvedený v súbore `README.txt` priloženého k aplikácii[5].

[3] http://www.freedesktop.org/wiki/Software/dbus

[4] http://qt.digia.com/

[5] aplikácia je voľne dostupná na stiahnutie na adrese http://medusa.fit.vutbr.cz/public/data/datasets/gestures/gesture_recognition.zip

# Kapitola 6

# Testovanie a výsledky

## 6.1 Detekcia ruky v obraze

Na natrénovanie detektoru som vytvoril vlastnú datovú sadu[1] obsahujúcu celkovo 1577 obrázkov od 10 rôznych ľudí. Tieto obrázky boli vyhotovené bežnou webkamerou a to pri rôznych svetelných podmienkach – umelé alebo prirodzené osvetlenie s rôznou intenzitou a zdrojom svetla umiestneným prevažne priamo pred snímaným objektom. Ukážka takto vyhotovených snímok je na obr. 6.1.

Aby boli tieto obrázky použiteľné na natrénovanie a otestovanie detektoru, z každého z nich som ručne „vystrihol“ štvorcovú oblasť obsahujúcu ruku. Takto novo vytvorené obrázky som náhodným výberom rozdelil do dvoch sád, trénovacej a testovacej, a pre každý obrázok som vytvoril jeho kópiu s pridaným gaussovským šumom. Počet obrázkov v trénovacej sade je teda 1532 a v testovacej 1610.

Ako pozadie som využil niekoľko stoviek vlastných obrázkov, voľne dostupnú dátovú sadu Caltech256[2], databázu osôb INRIA[3] a datovú sadu z VOC Challenge[4]. Z týchto sád boli odstránené obrázky obsahujúce ruku, čím vznikla databáza o veľkosti 18 082 obrázkov, z toho 4850 bolo použitých na natrénovanie a 13 232 na testovanie detektoru.

(a) (b) (c)

Obrázek 6.1: Ukážka vyhotovených snímok: (a) prirodzené svetlo, (b) prirodzené svetlo so zdrojom vľavo od objektu, (c) umelé svetlo.

[1] dostupná na http://medusa.fit.vutbr.cz/public/data/datasets/gestures/hand_dataset.tar, veľkosť 21 MiB

[2] http://www.vision.caltech.edu/Image_Datasets/Caltech256/

[3] http://pascal.inrialpes.fr/data/human/

[4] http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/index.html

Obrázek 6.2: Časť obrázkov použitých na natrénovanie detektoru.

### 6.1.1 Trénovanie detektoru

Pred samotným trénovaním bol pomocou nástroja `opencv_createsamples`, dostupného z OpenCV, z každého obrázku vytvorený binárny súbor obsahujúci 6 kópií tohoto obrázku. Každá kópia bola zmenšená na veľkosť $32 \times 32$ pixelov, prevedená do šedotónových farieb, boli na ňu aplikované rotácie okolo osí $x$, $y$ a $z$ s náhodnou veľkosťou uhlu z rozsahu $-14°$-$14°$ a hodnota všetkých pixelov bola zmenená pričítaním náhodnej hodnoty z rozsahu $-20$-$20$. Tieto súbory boli spojené do jedného a výsledný súbor bol použitý ako pozitívna sada na natrénovanie detektoru. Na obr. 6.2 je ukážka týchto obrázkov bez aplikovaných rotácií.

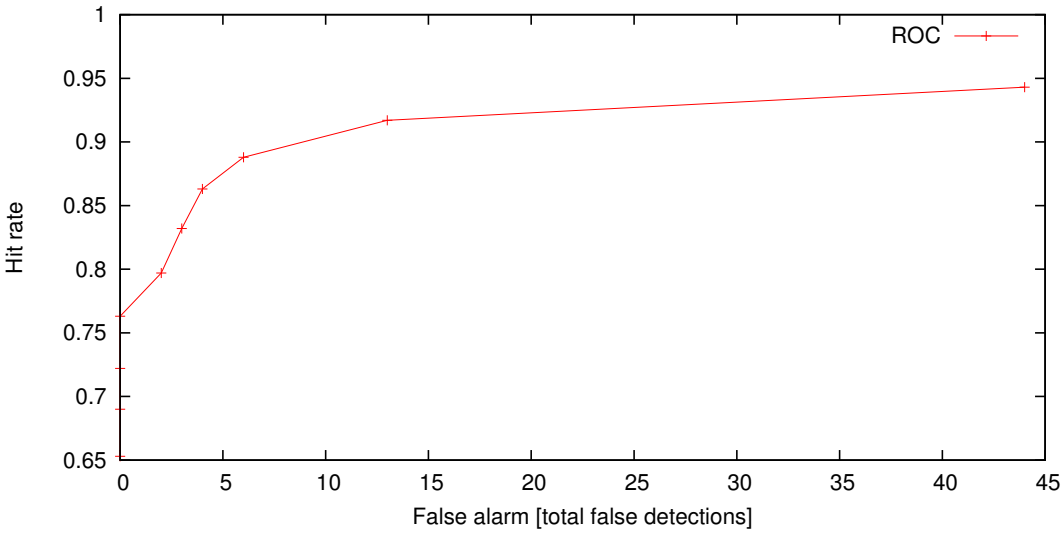
Na natrénovanie som využil nástroj `opencv_traincascade`, taktiež prítomný v OpenCV. Hodnoty detection rate a false alarm rate pre každý stupeň kaskády som nastavil na 0.995 resp. 0.45. Teoretická úspešňosť vybudovaného detektoru tak je $0.995^{18} \approx 0.91$ s hodnotou false alarm rate $0.45^{18} = 0.57 * 10^{-6}$. Celkový počet vyhodnocovaných LBP príznakov je 121. Výstupom trénovania je súbor `cascade.xml` obsahujúci informácie o počte klasifikátorov v kaskáde a použitých príznakoch. Informácie z tohoto súboru sú potom použité na klasifikáciu neznámych vzoriek.

### 6.1.2 Testovanie detektoru

Detektor bol testovaný na obrázkoch pozadia o veľkosti $320 \times 240$. Na každý z týchto 13 232 obrázkov bol na náhodnú pozíciu vložený obrázok ruky z testovacej sady s dĺžkou hrany 32-128 pixelov. Na obrázky rúk boli aplikované rovnaké transformácie ako pri trénovaní, viď obr. 6.3.

Detekcia objektu je založená na prístupe sliding window, kde sa detekčné okno posúva obrazom pixel po pixeli v rôznych veľkostiach. Objekt teda môže byť detekovaný i na pozíci-

Obrázek 6.3: Ukážka obrázkov použitých na testovanie detektoru.




Obrázek 6.4: ROC detektoru rúk. Hodnoty boli získané zvyšovaním počtu potrebných susedných detekcí. Testované na 13 232 obrázkoch veľkosti $320 \times 240$ px, na každom umiestnený práve jeden obrázok ruky.

ách susediacich s jeho skutočnou pozíciou. Tieto detekcie sú potom zlúčené do jednej, čoho výsledkom je poloha objektu v obraze. Minimálny potrebný počet detekcií na výslednú detekciu je v použitej klasifikačnej funkcii zároveň jediný parameter umožňujúci priamo ovplyvňovať hit rate/false alarm detektoru. ROC (Receiver Operating Curve) na obr. 6.4 bola podobne ako v [12], skonštruovaná postupným zvyšovaním tohoto parametru od 1 do 10. Hit rate je získaný ako pomer správnych detekcií k celkovému počtu rúk na obrázkoch, a keďže nie je k dispozícii skutočný počet susedných detekcií objektu, false alarmu vyjadruje skutočný počet falošných detekcií.

### 6.1.3 Zhodnotenie dosiahnutých výsledkov

Ako je vidieť na ROC, detektor dosahuje v syntetických testoch mimoriadne dobré výsledky, kde i pri minimálnom počte susedných detekcií 10, dokáže úspešne rozpoznať viac ako 60% rúk – a to pri nulovej chybovosti. Počet potrebných susedných detekcií som vo výslednom systéme nastavil na hodnotu 2, pri ktorej je dostatočne vysoká úspešnosť (92%) a zároveň

Obrázek 6.5: Desať náhodne vybraných gest každého typu. Posledný riadok obsahuje náhodné gestá.

zanedbateľný počet falošných detekcií – 13.

Nízky počet falošných detekcií sa potvrdil aj v reálnych experimentoch, úspešnosť detekcie je však silne závislá od smeru dopadu svetla. Zatiaľ čo pri priamom osvetlení funguje podľa očakávaní, pri silnejšom osvetlení so zväčšujúcim sa uhlom dopadu, jeho schopnosť detekcie výrazne klesá. Príčinou tohoto správania je vysoký kontrast medzi nasvietenou a nenasvietenou časťou ruky. Ďalší problém pri detekcii nastáva u ľudí s výrazne odlišnými obrysmi rúk, kedy ruka nebola v niekoľkých prípadoch rozpoznaná ani pri ideálnom osvetlení. Oba tieto problémy by ale bolo možné vyriešiť získaním dát od väčšieho množstva ľudí, a to za rozmanitejších svetelných podmienok.

## 6.2 Rozpoznanie gest

Podobne ako u detektoru, i tu som vytvoril datové sady[5] pre šesť použitých gest. Každá z týchto šiestich sád obsahuje 195-249 prevední jedného gesta od desiatich rôznych ľudí. Dĺžka trvania jedného gesta sa pohybuje najčastejšie v rozpätí 40-80 snímok (maximum je 120).

Okrem týchto šiestich sád, bola vytvorená i špeciálna sada obsahujúca náhodné gestá (225 prevedení) určená na zisťovanie false alarmu pre rôzne konfigurácie HMM. Pri jej získavaní bolo užívateľom povedané, aby „kreslili“ ľubovoľné vzory, nepripomínajúce ani jedno z uvedených gest. Niekoľko prevedení každého gesta je znázornených na obr. 6.5

Získané gestá boli uložené do textových súborov ako vektory dvojíc $x$, $y$ reprezentujúce stredy bounding boxov, s ktorými boli následne prevádzané testy.

### 6.2.1 Testovanie modelov a dosiahnuté výsledky

Testovanými parametrami u skrytých markovovych modelov, boli všetky kombinácie počtu stavov (6, 8,. . . , 14) a počtu emisií/symbolov v codebooku (15, 25, . . . , 65). Pre výber najvhodnejších parametrov som využil crossvalidačné testovanie nasledujúcim spôsobom.

[5] dostupné na http://medusa.fit.vutbr.cz/public/data/datasets/gestures/gestures.tar, veľkosť 25.6 GiB

Každú sadu gest som náhodným výberom rozdelil do piatich disjunktných množín. Pomocou každej z týchto množín som natrénoval jeden model, pričom jeho úspešnosť bola otestovaná na gestách zo zvyšných štyroch množín. Výsledná úspešnosť ($tpr$ – true positive rate) je potom daná ako pomer všetkých správne rozpoznaných gest ku všetkým gestám. Modely boli testované i na náhodných gestách, kde bol false positive rate ($fpr$) získaný ako pomer náhodných gest rozpoznaných nejakým modelom, ku všetkým náhodným gestám.

Z týchto dvoch hodnôt bola vyrátaná hodnota *F-measure* ako

$$P = \frac{tpr}{tpr + fpr}, \tag{6.1}$$

$$R = \frac{tpr}{tpr + fnr}, \tag{6.2}$$

$$F = 2 \cdot \frac{P \cdot R}{P + R}, \tag{6.3}$$

kde *Precision* $P$ vyjadruje dôveryhodnosť rozpoznania a *Recall* $R$ celkovú úspešnosť rozpoznania ($tpr$). Čím je hodnota F-measure vyššia, tým je rozpoznanie spoľahlivejšie.

V tabuľkách 6.1 a 6.2 sú zobrazené výsledky testov bez aplikovania prahu pre klasifikáciu, resp. s prahom $-200$ (log. pravdepodobnosť sekvencie pre každý model). Klasifikovaním gest s pravdepodobnosťou menšou než $-200$, ako nerozpoznaných, bolo u väčšiny kombinácií parametrov dosiahnuté zvýšenie hodnoty F-measure, pričom najvyššia hodnota bola dosiahnutá pri parametroch $N = 10$ a $M = 35$. Úspešnosť u crossvalidácie tu dosahuje takmer 91% a false alarm rate 6.6%. S týmito parametrami boli natrénované i modely použité v aplikácii a to využitím všetkých získaných sekvencií.

V tabuľkách 6.3 a 6.4 sú zobrazené úspešnosti rozpoznávania jednotlivých modelov pri vyššie uvedených parametroch. Modely a gestá nimi rozpoznávané sú číslované tak ako na obr. 5.3 v kapitole 5. I tieto hodnoty boli získané crossvalidáciou, tentokrát však rozdelením sád na dve časti. Prahovaním pravdepodobností klesol false alarm rate o takmer 17% (24.4 vs. 7.6), celková úspešnosť rozpoznania ale len o niečo vyše 2% (95.4 vs. 93.1). Za povšimnutie stojí taktiež nulový, alebo takmer nulový počet gest nesprávne rozpoznaných iným modelom.

Vysoká úspešnosť rozpoznávania gest sa potvrdila i v reálnych testoch.

| Emisie / Stavy | 15 | 25 | 35 | 45 | 55 | 65 |
|---|---|---|---|---|---|---|
| 6 | 0.717 | 0.768 | 0.786 | 0.794 | 0.809 | 0.822 |
| 8 | 0.789 | 0.838 | 0.855 | 0.855 | 0.861 | 0.876 |
| 10 | 0.824 | 0.862 | 0.878 | 0.884 | 0.892 | 0.888 |
| 12 | 0.862 | 0.886 | 0.891 | 0.888 | 0.894 | 0.901 |
| 14 | 0.814 | 0.872 | 0.885 | 0.878 | 0.883 | 0.87 |

Tabulka 6.1: Hodnoty F-measure pre rôzne parametre HMM, bez aplikovania prahu, získané crossvalidáciou rozdelením do piatich častí.

| Emisie / Stavy | 15 | 25 | 35 | 45 | 55 | 65 |
|---|---|---|---|---|---|---|
| 6 | 0.861 | 0.882 | 0.861 | 0.845 | 0.83 | 0.826 |
| 8 | 0.89 | 0.91 | 0.911 | 0.89 | 0.876 | 0.862 |
| 10 | 0.903 | 0.918 | **0.923** | 0.909 | 0.894 | 0.874 |
| 12 | 0.912 | 0.92 | 0.918 | 0.902 | 0.884 | 0.871 |
| 14 | 0.842 | 0.895 | 0.899 | 0.871 | 0.847 | 0.82 |

Tabulka 6.2: Hodnoty F-measure pre rôzne parametre HMM, aplikovaný prah $-200$, získané crossvalidáciou rozdelením do piatich častí.

| Modely / Gestá | M 01 | M 02 | M 03 | M 04 | M 05 | M 06 | Nerozpoznané |
|---|---|---|---|---|---|---|---|
| G 01 | **0.96** | 0.005 | 0.0 | 0.0 | 0.0 | 0.03 | 0.005 |
| G 02 | 0.0 | **0.939** | 0.0 | 0.01 | 0.0 | 0.02 | 0.03 |
| G 03 | 0.0 | 0.0 | **0.969** | 0.005 | 0.0 | 0.0 | 0.026 |
| G 04 | 0.0 | 0.0 | 0.005 | **0.99** | 0.0 | 0.0 | 0.005 |
| G 05 | 0.01 | 0.0 | 0.0 | 0.0 | **0.951** | 0.0 | 0.039 |
| G 06 | 0.06 | 0.012 | 0.0 | 0.0 | 0.0 | **0.912** | 0.016 |
| Náhodné | 0.044 | 0.031 | 0.049 | 0.058 | 0.013 | 0.049 | **0.756** |

Tabulka 6.3: Úspešnosť rozpoznávania gest jednotlivými modelmi pri $N = 10$, $M = 35$. Datové sady boli rozdelené na dve časti, bez aplikovania prahu.

| Modely / Gestá | M 01 | M 02 | M 03 | M 04 | M 05 | M 06 | Nerozpoznané |
|---|---|---|---|---|---|---|---|
| G 01 | **0.955** | 0.005 | 0.0 | 0.0 | 0.0 | 0.02 | 0.02 |
| G 02 | 0.0 | **0.929** | 0.0 | 0.01 | 0.0 | 0.01 | 0.051 |
| G 03 | 0.0 | 0.0 | **0.954** | 0.0 | 0.0 | 0.0 | 0.046 |
| G 04 | 0.0 | 0.0 | 0.0 | **0.981** | 0.0 | 0.0 | 0.019 |
| G 05 | 0.01 | 0.0 | 0.0 | 0.0 | **0.883** | 0.0 | 0.107 |
| G 06 | 0.052 | 0.012 | 0.0 | 0.0 | 0.0 | **0.888** | 0.048 |
| Náhodné | 0.013 | 0.004 | 0.013 | 0.022 | 0.009 | 0.013 | **0.924** |

Tabulka 6.4: Úspešnosť rozpoznávania gest jednotlivými modelmi pri $N = 10$ a $M = 35$. Datové sady boli rozdelené na dve časti, na klasifikáciu aplikovaný prah $-200$.

# Kapitola 7

# Záver

Cieľom tejto bakalárskej práce bolo vytvoriť systém, umožňujúci ovládanie počítača cez webkameru a to pomocou jednoduchých gest ruky. Pre jeho splnenie bolo potrebné naštudovať metódy používané na detekciu objektu v obraze, sledovanie objektu a rozpoznávanie vzorov.

Ako detektor bola využitá kaskáda boostovaných klasifikátorov. Na jej natrénovanie bola vytvorená sada obsahujúca niekoľko stoviek obrázkov rúk od desiatich rôznych ľudí. Takto natrénovaný detektor dosahuje výborné výsledky pri syntetických testoch, jeho reálna úspešnosť je ale výrazne závislá od svetelných podmienok okolia. Ako tracker bol využitý systém TLD, umožňujúci spoľahlivé sledovanie ľubovoľného objektu po jeho počiatočnej inicializácii. Rozpoznanie vzorov/gest bolo realizované použitím Skrytých markovovych modelov. Na ich natrénovanie bola vytvorená datová sada obsahujúca niekoľko desiatok prevedení každého gesta, získaná taktiež od desiatich osôb. Úspešnosť rozpoznávania gest je podobne ako u detektoru veľmi vysoká a potvrdila sa i v reálnych testoch.

Tieto tri komponenty boli implementované použitím voľne dostupných knižníc a nástrojov a ich prepojením vznikol výsledný systém. Vďaka pripojeniu na rozhranie D-Bus je tak pomocou neho možné ovládať teoreticky ľubovoľný program v počítači pripojený na toto rozhranie. Výstupom tejto práce je teda program, pomocou ktorého je možné šiestimi jednoduchými gestami ovládať prehliadač dokumentov a multimediálny prehrávač. Systém podporuje iba ovládanie pravou rukou a funguje pri prirodzenom i umelom osvetlení. Vytvorený program je spolu s nazhromaždenými dátami voľne dostupný k stiahnutiu.

Pre ďalší vyvoj systému by bolo potrebné získať viac dát na natrénovanie detektoru a zároveň umožniť ovládanie systému i ľavou rukou. Ďalším možným rozšírením by bola možnosť ovládania viacerých programov súčasne s vhodným mechanizmom prepínania medzi nimi. Takto vybudovaný systém však nie je obmedzený len na použitie v osobných počítačoch. Po jeho integrácii do mobilného telefónu alebo fotoaparátu by mohol slúžiť napríklad ako spúšť, u počítačov typu Raspberry Pi zasa ako „diaľkový ovládač“ na rôznu domácu elektroniku.

# Literatura


[1] Alsabti, K.; Ranka, S.; Singh, V.: An efficient k-means clustering algorithm. 1997.

[2] Bouguet, J.-Y.: Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 2001.

[3] Chang, F.; Chen, C.-J.; Lu, C.-J.: A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 2004.

[4] Crow, F. C.: Summed-area tables for texture mapping. In *ACM SIGGRAPH Computer Graphics*, 1984.

[5] Ferreira, A.: Survey on Boosting algorithms for supervised and semi-supervised learning. *Institute of Telecommunications*, 2007.

[6] Freund, Y.; Schapire, R. E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 1997.

[7] Hu, J.; Brown, M. K.; Turin, W.: HMM based online handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1996.

[8] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Forward-backward error: Automatic detection of tracking failures. In *Pattern Recognition (ICPR), 20th International Conference on*, 2010.

[9] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2012.

[10] Kunaver, M.; Tasic, J.: Image feature extraction-an overview. In *Computer as a Tool, 2005. EUROCON 2005. The International Conference on*, 2005.

[11] Liao, S.; Zhu, X.; Lei, Z.; aj.: Learning multi-scale block local binary patterns for face recognition. In *Advances in Biometrics*, 2007.

[12] Lienhart, R.; Kuranov, A.; Pisarevsky, V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Pattern Recognition*, 2003.

[13] Lucas, B. D.; Kanade, T.; aj.: An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.

[14] Nebehay, G.: *Robust Object Tracking Based on Tracking-Learning-Detection.* Dizertační práce, Faculty of Informatics, TU Vienna, 2012.

[15] Ojala, T.; Pietikäinen, M.; Harwood, D.: A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 1996.

[16] Ozuysal, M.; Fua, P.; Lepetit, V.: Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition, 2007. IEEE Conference on*, 2007.

[17] Rabiner, L. R.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

[18] Radke, R. J.; Andra, S.; Al-Kofahi, O.; aj.: Image change detection algorithms: a systematic survey. *Image Processing, IEEE Transactions on*, 2005.

[19] Stojmenovic, M.: Real time machine learning based car detection in images with fast training. *Machine Vision and Applications*, 2006.

[20] Stojmenovic, M.: Algorithms for real-time object detection in images. *Handbook of Applied Algorithms: Solving Scientific, Engineering, and Practical Problems*, 2007.

[21] Viola, P.; Jones, M. J.: Robust real-time face detection. *International journal of computer vision*, 2004.

[22] Yoon, H.-S.; Soh, J.; Bae, Y. J.; aj.: Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 2001.

# Příloha A

# Obsah CD

```
-+-- technicka_sprava.pdf                Technická správa (tento súbor).
 |
 +-- plagat.pdf                          Plagát.
 |
 +-- video.mp4                           Video obsahujúce ukážku behu aplikácie.
 |
 +-- src --+-- program_src.zip           Zdrojové kódy aplikácie.
           |
           +-- technicka_sprava_src.zip  Zdrojové kódy technickej správy (LaTeX).
```